UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/708,159	11/08/2000	Toshiaki Yasue	JP919990097US1	1032

7590 01/26/2006

William A Kinnaman Jr
IBM Corporation - MS P386
2455 South Road
Poughkeepsie, NY 12601

EXAMINER	
RUTTEN, JAMES D	

ART UNIT	PAPER NUMBER
2192	

DATE MAILED: 01/26/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/708,159	YASUE ET AL.	
	Examiner	Art Unit	
	J. Derek Rutten	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *14 October 2005*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-10* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-10* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to Applicant's amendment dated 10/14/2005, responding to the 7/14/2005 Office action provided in the rejection of claims 1-10, wherein claims 1, 4, 5, 7, 8, and 10 have been amended. Claims 1-10 remain pending in the application and have been fully considered by the examiner.

2. Applicant's arguments, see pages 12-14, filed 10/14/05, with respect to the rejection(s) of claim(s) 1-10 under 35 U.S.C. § 103 have been fully considered and are persuasive in part. Therefore, the rejection has been withdrawn. However, upon further consideration, a new ground(s) of rejection is made in view of U.S. Patent 5,530,866 to Koblenz et al.

Response to Arguments

3. In response to the 35 U.S.C. § 112, first paragraph, rejection of claims 1-10, Applicant essentially argues (see item 1 on pages 6-8 of the response filed 10/14/05) that the terms "loop" and "loop process" refer to any loop in the compiled code process, and not just a main program loop. Applicant asserts that those of ordinary skill would not restrict their interpretation of the terms to the application of a main program loop. In the context of the claim these terms are used in relation to a method that is transferred "from an interpreter process to a loop process of a compiled code process". The recitation of "interpreter process" in contrast to a "compiled code process" implies that there are two processes executing in different address spaces. The term "process" is a broad term that could be used in the context of program execution to describe a

running instance of a program. The online encyclopedia Wikipedia (www.wikipedia.org - 12/30/05) gives one description of a process as "a running instance of a program, including all variables and other states". However, in light of Applicant's comments, it appears that the word "process" as used in the claims is not meant to describe a particular running instance of a program per se, but rather as a word that describes a flow graph. In this case, an "interpreter process" would be descriptive of a portion of the flow graph relating to execution of interpreted code. Likewise, a "compiled code process" is simply descriptive of a portion of a flow graph related to the execution of compiled code. Further, a "loop process of a compiled code process" is simply descriptive of a portion of the flow graph related to a loop that exists within the portion of the flow graph related to the execution of compiled code. As such, Applicant's arguments are convincing, and the rejection is withdrawn.

4. Applicant's arguments regarding the rejection of claims 4 and 8-10 under 35 U.S.C. § 112 second paragraph are convincing. Therefore, the rejection is withdrawn.

5. Applicant's amendments to independent claims 1, 4, 5, 7, 8, and 10 have overcome the rejection of claims 1-10 under 35 U.S.C. § 101. Therefore, this rejection has been withdrawn.

6. Applicant's arguments (pages 11 - 13) regarding the 35 U.S.C. § 103 rejection of claims 1-10 over Aho et al. in view of Bak are convincing. Applicant's arguments essentially point out that the invention is concerned not with the movement of code, but with the movement of transfer points, and that the Examiner's citation of Aho is not related to transfer points, but rather deals with the techniques of code motion (page 12 paragraph 1). Therefore, this rejection is withdrawn.

7. Applicant's arguments (pages 13-14) regarding the 35 U.S.C. § 103 rejection of claims 1-10 over Aho et al. in view of Bak are not convincing. Applicant essentially argues that Aho Fig. 10.49(a) (Aho page 668) does not depict transfer points (10/14/05 page 14 paragraph 1). Applicant has previously argued that branch instructions may be interpreted broadly as transfer points (see 03/04/05 page 5 paragraph 2). Fig. 10.49(a) shows a flow graph made of nodes and edges, which when implemented in code must include branches. Applicant has also previously argued that transfer points are represented by special edges in a flow graph (4/1/04 page 5 last paragraph). Fig. 10.49(a) has nodes 1-3 with an edge flowing from node 1 to node 2, and an edge flowing from node 1 to node 3. At least one of these edges must be considered as an entry point to a loop while the other might be interpreted as a transfer point. Aho further describes this figure in terms of node splitting, and a similar figure 10.57 is presented on page 680 in terms of nonreducible flow graphs. Applicant has previously argued that the presence of transfer points render loops irreducible (10/14/05 page 10 paragraph 2). Aho's description of nonreducibility includes a flow graph with back edges whose heads do not dominate their tails (Aho pages 606-607). This also appears to occur with applicant's discussion of transfer points as depicted in the figure on page 11. Thus, Applicant's arguments are not persuasive, since Aho's Fig. 10.49(a) could be interpreted as depicting transfer points to a loop.

Claim Rejections - 35 USC § 112

8. The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

9. Claims 1-10 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the enablement requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to enable one skilled in the art to which it pertains, or with which it is most nearly connected, to make and/or use the invention. The claims are concerned with moving and/or copying transfer points. The specification describes transfer points in a variety of contexts:

- A transfer point designated by the interpreter is defined as the first in the BB. (page 10 lines 2-3)
- Transfer point: a point whereat program execution is transferred from the Java interpreter 124 to the JIT compiler 126. (page 14 lines 28-29)
- Transfer bwd jump: a rearward branching command used by the Java interpreter 124 to detect a transfer point. (page 15 lines 1-2)

However, these descriptions do not enable one of ordinary skill to locate, copy, and/or move a transfer point. Applicant has previously asserted that one of ordinary skill would know how to manipulate transfer points (see 4/1/04 page 7), and this argument was previously found to be convincing. However, Applicant's current arguments that movement of transfer points do not involve movement of code (see 10/14/05 page 12), and that Aho's depiction of node splitting does not show transfer points to a flow graph (10/14/05 page 13 continuing to page 14) has rendered the previous argument unconvincing. Since it is not clear exactly what a transfer point is, and the originally filed specification does not provide a clear description of how to find, move, or copy one, one of ordinary skill would not be able to make and/or use the invention. For the purpose of further examination, transfer points will be interpreted broadly in terms of edges in a flow graph which may be representative of any of a variety of control flow instructions present in code.

10. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

11. Claims 1-10 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

12. Claims 1-10 are rejected under 35 U.S.C. 112, second paragraph, as failing to set forth the subject matter which applicants regard as their invention. Evidence that claims 1-10 fail to correspond in scope with that which applicants regard as the invention can be found in the replies filed 4/1/04 on page 7, 3/4/05 page 5, and 10/14/05 page 11. In these papers, applicant has stated:

> ...To compile a method including transfer points, each transfer point is represented by a special edge linked from the entry node to the transfer point on the control flow graph of the method. [4/1/04 page 5 last paragraph]
> ...
> Except for applicants' inventive steps of moving transfer points and copying regions of a loop, **all of this – including the notions of transfer points**, dominating nodes and irreducible loops – is well known in the art, as shown, for example by the discussion in **Section 10.4 (pages 602-608) of Aho et al.**, Compilers: Principles, Techniques, and Tools (1986), cited by the Examiner. [see 4/1/04 page 7 – emphasis added]
> ...
> The Examiner argues (page 3, ¶ 5) that lines 150 and 200 of the code listing shown in Fig. 6 (corresponding generally to node B in Fig. 9A) constitute such transfer points. However, line 150 contains a conditional branch to line 250, within the original loop (lines 100-400), while line 200 contains an unconditional branch to line 300, also within the original loop. So **if these lines are, broadly speaking, "transfer points", they are transfer points within the loop rather than transfer points into the loop (or entry points, using O'Brien's terminology) as claimed by applicants**. Indeed, if lines 150 and 200 were such entry points, then, given the existing entry point at line 100, the loop would no longer be a single-entry strongly connected region (SESCR) and would therefore be deemed ineligible for processing (col. 4, lines 10-17). [see 3/4/05 page 5 – emphasis added]
> ...
> The code motion discussed in the excerpts cited above deal with the movement of code to outside of a loop (in particular, before the loop) if the values generated are the same for any iteration of the loop. **This aspect of applicant's invention, however, does not involve the movement of code; rather, it involves the movement of a transfer point to a specified position in a loop**. So it is not seen how Aho's discussion of code motion is even relevant here or how it would "inform" a person of ordinary skill in the art. [see 10/14/05 page 12 – emphasis added]
> ...

> ...More relevantly, the Examiner points to the node-splitting example shown in Fig. 10.49 on page 668 [referring to Aho]. There, the node 2 of Fig. 10.49(a) is split into the pair of nodes 2a and 2b shown in Fig. 10.49(b), permitting the further simplifications shown in Fig. 10.49(c) and (d). However, while this shows copying of a node, **there are no depicted entry points or transfer points to the flow graph formed by the nodes illustrated in Fig. 10.49(a)**... [see 10/14/05 page 13 continuing on page 14 – emphasis added]

In summary: (1) Transfer points are represented by special edges in a flow graph; (2) Aho discusses transfer points; (3) Branch statements might be broadly related to transfer points; (4) The movement of transfer points does not involve the movement of code; and (5) Aho's flow graph edges of Fig. 10.49(a) do not depict transfer points to a flow graph. These statements indicate that the invention is different from what is defined in the claims because the manipulation of transfer points in terms of "copying code...to a point that post-dominates...said one or more transfer points" (claim 1 lines 9-12) implies that transfer points are present in code, and that movement of a transfer point would necessarily involve movement of code. However, as cited above, Applicant has essentially stated that moving transfer points does not involve movement of code (10/14/05 page 12). Further, the claims recite "...code that includes one or more transfer points at which program execution is transferred from the interpreter process to said loop process...". Note that, as explained above, the term "process" is used in the context of a flow graph. Applicant has stated that Aho's depiction of node splitting in Fig. 10.49(a) does not contain "transfer points to the flow graph" (10/14/05 page 13 continuing on page 14). However, the depiction in Fig. 10.49(a) shows an edge from node 1 to node 2, and also an edge from node 1 to node 3, at least one of which must be implemented using some type of branch statement, and which would appear to satisfy the claim language for "transfer points...to said loop process". Thus, the broad scope of the claim language does not appear to correspond with applicant's invention as presently argued on pages 12-14 of the 10/14/05 reply. As stated above,

for the purpose of further examination, transfer points will be interpreted broadly in terms of edges in a flow graph which may be representative of any of a variety of control flow instructions present in code.

Claim Rejections - 35 USC § 103

13. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

14. Claims 1-10 are rejected under 35 U.S.C. 103(a) as being unpatentable over prior art of record "Compilers: Principles, Techniques, and Tools" by Aho et al. (hereinafter "Aho") in view of prior art of record U.S. Patent 6,513,156 to Bak et al. (hereinafter referred to as "Bak"), further in view of prior art of record "Compiler Transformations for High-Performance Computing" by Bacon et al. (hereinafter referred to as "Bacon"), further in view of U.S. Patent 5,530,866 to Koblenz et al. (hereinafter "Koblenz").

In regard to claim 1, Aho discloses:

A program execution method ... comprising the steps of: optimizing the loop process, said optimizing step including the steps of: copying code from the top of the loop process to a point that post-dominates said top of said loop process and said one or more transfer points to a location immediately preceding said loop process if said transfer

points are located inside said loop process; Aho discusses code optimizations that benefit from the reducibility of flow graphs containing loops in terms of using intervals, interval graphs, and node splitting (see pages 664-668). In particular, node splitting is a technique that is used to produce a limit flow graph of a single node (page 666, last paragraph). Additional nodes are created that precede the original node. Fig. 10.49 shows a flow graph with 3 nodes with edges from 1 to 2, 1 to 3, 2 to 3, and 3 to 2. The loop of nodes 2 and 3 show two incoming edges. While one edge must be regarded as the entry point, the other must be regarded as a transfer point, thereby making the loop irreducible. Node 2 is split into nodes 2a and 2b. When combined with node 1, node 2a now precedes the loop that is formed between nodes 2b and 3. Further description is found on pages 679-680, which describes duplicating a region that represents a node, and placing that region in a location preceding the node. This process is shown in Fig. 10.57 on page 680, which shows a copied node containing a transfer point that immediately precedes the loop process.

Aho further discloses common subexpression elimination (Fig. 10.7 on page 593), and code motion (page 596).

Aho does not expressly disclose: moving transfer points to the top of a loop process; transferring a method from an interpreter process to a compiled code process; storing information for generating recalculation code for specific transfer points; performing a recalculation during a transfer process; or privatization.

However, in an analogous environment, Koblenz teaches: *moving said one or more transfer points to the top of a loop process if they can be moved there without a problem occurring*; See Koblenz column 8 lines 23-28:

> Irreducible loops do not exhibit a loop top; however, all basic blocks in an irreducible loop that are reached by a forward control flow edge from a basic block outside the loop can be combined into a single summary loop top in constructing the tile tree. This summary node will dominate every basic block in the loop.

Also, in an analogous environment, Bak teaches: *transferring, from an interpreter process to a compiled code process, a method that is currently being executed for code that includes a plurality of transfer points at which program execution is transferred from the interpreter process to the compiled code process <via one of said transfer points>*. See column 2 lines 40-45:

> the hybrid virtual and native machine instructions may be easily transformed back to the original virtual machine instructions, and the flexibility of compiling only certain portions of a function into native machine instructions allows for better optimization of the execution of the function.

storing information for generating recalculation code for one or more specific transfer points See column 2 line 65 – column 3 line 1:

> A copy of a selected virtual machine instruction at a beginning of the portion of the function is **stored** and a back pointer to a location of the selected virtual machine instruction is also stored.

and performing a recalculation during a transfer process See column 3 lines 1-5:

> The selected virtual machine instruction is overwritten with a new virtual machine **instruction that specifies execution** of the native machine instructions so that the function includes both virtual and native machine instructions.

Bak is generally concerned with mixed execution of interpreted and compiled code sequences where the compiled code process is referred to as a "snippet", and transfer points to the compiled code process are indicated with a "go--native" instruction (column 6 lines 31-35 and column 7 lines 28-37).

Also in an analogous environment, Bacon teaches: *privatization* See page 395 Section 7.1.3:

> When a scalar is used within a loop solely as a scratch variable, each processor can be given a private copy so the use of the scalar need not involve any communication.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Koblenz' moving of transfer points with Bacon's optimizations with Bak's mixed mode interpreter in Aho's code optimizer. One of ordinary skill would have been motivated to remove transfer points from a loop in order to make them reducible since many optimizations depend on reducibility (Aho page 607). Further, one would have been motivated to transfer the execution of an interpreted loop to natively compiled instructions since native code executes more quickly than interpreted code.

As per claim 2, the above rejection of claim 1 is incorporated. Aho does not expressly disclose choosing transfer points for transferring from interpreted mode to compiled mode execution.

However, Bak teaches *defining as a new transfer point, a point from said interpreter process to said compiled code process whereat, when said method that is currently being executed is replaced, the execution speed is increased compared with when said method is not replaced* (column 6 line 61 – column 7 line 5).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Bak's selection of transfer points in O'Brien's code optimizer.

One of ordinary skill would have been motivated to improve code so that a program will execute in less time.

As per claim 3, the above rejections of claims 1 and 2 are incorporated. Aho does not expressly disclose generating, storing, or employing information for transferring execution from interpreted to compiled execution.

However, Bak teaches:

generating information required to perform a transfer from said interpreter process to said compiled code process (column 7 lines 28-40)*; and*

storing said generated information while correlating said generated information with said transfer points (column 7 lines 28-40 as cited above),

wherein, at said recalculation step, said information stored for said transfer points is employed (column 7 lines 63-67).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Bak's transfer information with O'Brien's code optimizer. One of ordinary skill would have been motivated to enable the transfer of interpreted execution to natively compiled execution, which is necessarily supported by information regarding the location of code, to increase the speed of a program.

As per claim 4, Aho does not expressly disclose a program storage device.

However, Bak teaches the use of a program storage device to hold program instructions (column 4 lines 46-50).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Bak's program storage device with O'Brien's code optimizer. One of ordinary skill would have been motivated to store copies of a program on media that enables the distribution of the program to colleagues or customers.

All further limitations have been addressed in the above rejection of claim 1.

15. Claim 5 is rejected under 35 U.S.C. 103(a) as being unpatentable over Bak in view of Koblenz.

16. Claims 5, 6, 8, and 9 are rejected under 35 U.S.C. 103(a) as being unpatentable over Bak, in view of Koblenz and Aho.

In regard to claim 5, all limitations have been addressed in the above rejection of claim 1. It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Koblenz' movement of transfer points, and Aho's optimization with Bak's transfer of execution. One of ordinary skill would have been motivated to make a loop reducible in order to better optimize it (Bak column 2 lines 24-29, Aho first paragraph of section 10.9 on page 660, and Koblenz column 8 lines 15-23).

In regard to claim 6, all limitations have been addressed in the above rejection of claim 1. It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Aho's code copying with Bak's transfer process in order to facilitate the reducibility of a graph which would allow better optimization.

In regard to claim 8, all limitations have been addressed in the above rejections of claims 4 and 5.

In regard to claim 9, the above rejection of claim 8 is incorporated. All further limitations have been addressed in the above rejection of claim 1.

17. Claim 7 is rejected under 35 U.S.C. 103(a) as being unpatentable over Aho in view of Bak.

In regard to claim 7, all limitations have been addressed in the above rejection of claim 1.

In regard to claim 10, all limitations have been addressed in the above rejection of claims 1 and 4.

Conclusion

18. Any inquiry concerning this communication or earlier communications from the examiner should be directed to J. Derek Rutten whose telephone number is (571) 272-3703. The examiner can normally be reached on T-F 6:00 - 4:30.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

jdr

TUAN DAM
SUPERVISORY PATENT EXAMINER